

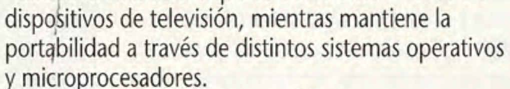
Sumario

Sun apuesta por su tecnología API Java TV



Sun Microsystems ha anunciado recientemente la incorporación de empresas clave relacionadas con la televisión digital al proceso de desarrollo de la API (*Application Programming Interface*) Java TV. Entre las empresas participantes se encuentran Philips, Sony, Toshiba, LG Electronics, OpenTV, Motorola, etc. Este grupo de trabajo, que incluye a Sun y a los líderes tecnológicos arriba mencionados, está desarrollando la API Java TV para que ésta pueda ser incorporada dentro de los estándares de la televisión digital.

Cualquier dispositivo o televisión digital que soporte la norma Java TV podrá ofrecer a sus usuarios, entre otras funcionalidades, una televisión con contenidos interactivos, Vídeo Bajo Demanda (VOD), Guías de Programación Electrónicas (EPGs) y ángulos de visión personalizables en eventos deportivos. La API está siendo diseñada para proporcionar acceso a datos y entretenimiento a través de un sofisticado control de los flujos digitales de vídeo, audio, aplicaciones y otros datos. API Java TV, además, proporcionará una plataforma software de acceso independiente a las características físicas que son comunes a todas los dispositivos de televisión, mientras mantiene la portabilidad a través de distintos sistemas operativos y microprocesadores.





Esta norma se ha creado con el fin de proveer una solución a las necesidades de fabricantes, operadores de comunicaciones y proveedores de contenido en su búsqueda de un estándar para proporcionar contenidos interactivos a sus clientes a través de conexiones cable, receptores vía satélite y televisiones digitales. La API Java TV pretende que los fabricantes puedan introducir nuevos productos que puedan fácilmente adaptarse fácilmente a nuevas formas de contenido interactivo a medida que éstos están disponibles en el mercado.

Para más información visitar la página: *http://www.sun.com/consumer-embedded/cover/java7v.html*

Cryo cambia de logo



La famosa desarrolladora de videojuegos francesa, Cryo Interactive, ha cambiado su logotipo. Esta compañía, productora de juegos de la talla de *Ring* o *Saga*, siempre a la cabeza en temas de diseño y estética, parece que ha terminado por considerar su insignia un tanto anticuada para los nuevos tiempos; es decir, que parece que anticipan un cambio de generación en el mundo lúdico y no han querido quedarse atrás.

Electronic Arts amplía su influencia

Electronic Arts, una de las compañías que más ha crecido en el mundo de los videojuegos en los últimos años, ha firmado una licencia para desarrollar juegos de Fórmula 1. Vamos, que en esta ocasión ha realizado un elegante adelantamiento en toda regla.

Por si esto fuera poco, ha firmado otro acuerdo, en esta ocasión con elo sello Looking Glass Studios para la distribución conjunta de videojuegos. *Flight Unlimited III* y *System Shock 2* serán los primeros programas que saldrán al mercado como consecuencia de este acuerdo.

Sin duda, Electronic Arts, se encuentra un momento de auge que le da un margen muy favorable para invertir en acuerdos, alianzas y nuevas firmas. No será extraño que, durante los próximos meses, oigamos más acerca de su estrategia para hacerse con una porción aun mayor del mercado.

No olvidemos que, hasta el momento, Electronic Arts tiene tratos con algunas de las compañías programadoras más importantes del sector, entre las que se cuentan casas como Firaxis (Alpha Centaury). la divisi8ón interactiva de Fox (Expediente X, Alien Vs. Predator), Bulfrog (Populous, The Begining), Jane's (F-15, World War II Fighters, Israeli Air Force), Accolade (Rerdline, SimCity 3000), LucasArts (Grim Fandango, Star Wars Rebellion), DreamWorks Interactive (Trespasser). Westwood Studios (Dune 2000), Cyan (Riven), Metr ó Goldwin Meyer (Wargames) o Hasbro (Axis & Allies) o su propio sello deportivo EA Sports. por mucho que se nos haya quedado en el tintero, se trata sin duda de una de las compañías ditribuidoras más fuertes de nuestro país. Y cada vez lo es más.



Dentro de nuestro CD-Rom de portada incluimos en esta ocasión el siguiente material relacionado con la sección Game Developer:

- **Los códigos fuente, con las texturas, los gráficos y el sonido de un excelente programa tridimensional disponible freeware, Golgotha.**

Autor: Paulo Tromp

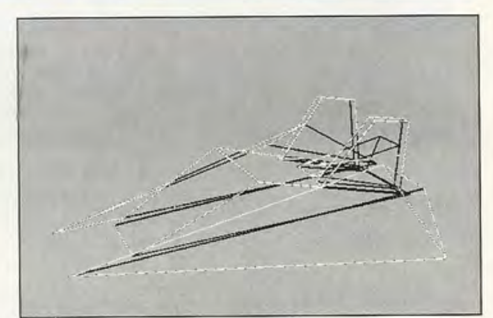
Modelado de objetos en baja poligonización

En esta entrega modelaremos una nave en baja poligonización y la texturizaremos y mapearemos de forma adecuada para poder ser utilizada, posteriormente, en un videojuego 3D.

Con la constante investigación y desarrollo de nuevas tarjetas aceleradoras para ordenador, la cantidad y calidad de los juegos que hacen uso de éstas va siendo cada vez mayor. El gran debate sobre qué API utilizar a la hora de desarrollar un engine esta servido: Glide, Direct3D, OpenGL... muchos programadores optan por el 2º debido a la compatibilidad con una amplia gama de tarjetas gráficas. OpenGL, sin embargo, resulta también una opción atractiva al ser más intuitivo que los otros. Las tarjetas 3DFX tuvieron un gran éxito con su Glide, acelerando incluso los APIs anteriormente citados. Todo ello, unido a su gran potencia de proceso, hizo que terminara instalándose en muchos hogares de adictos a los juegos tridimensionales.

OpenGL, sin embargo, resulta también una opción atractiva al ser mas intuitivo que los otros

Con las tarjetas aceleradoras que también renderizaban en modo ventana se hizo la oferta todavía mas atractiva. Poder jugar en altas resoluciones, con un gran número de polígonos, acelerando asimismo los gráficos de la mayoría de los programas de diseño, se hizo muy apetitoso. Siendo además el precio asequible para la mayoría de los bolsillos, no había duda alguna de que abrirían un hueco importante en el mercado informático.
Los grafistas han podido aprovechar enormemente la aparición de estos soportes. Anteriormente tenían muchas mas limitaciones a la hora de diseñar objetos 3D para ser utilizados por aplicaciones en tiempo real. Recurriendo a la aceleración por hardware es posible desarrollar juegos con mayor cantidad de polígonos y mayor calidad de texturas, consiguiendo asimismo una gran fluidez en las animaciones. De esta forma el grafista puede explotar mejor la creatividad que lleva dentro. El diseño de objetos y escenarios, para ser utilizados en imágenes y animaciones prerrenderizadas, difiere completamente del que se utiliza para un juego tridimensional en tiempo real. En el primer caso no hay limitación en cuanto al detalle de las mallas, del número de luces y texturas y de numerosos efectos especiales, mas la que impone el propio equipo. Todo esto será calculado por el programa de diseño y pasado a una imagen o animación legible por el juego. Sin embargo, los ordenadores personales de hoy en día no tienen la potencia suficiente como para poder realizar los cálculos necesarios para mover mallas densas en tiempo real con fluidez. Por todo esto, los objetos a utilizar tienen que tener una cantidad de polígonos relativamente pequeña.


FIGURA 2

3D Studio MAX es un programa muy utilizado en el diseño de gráficos para videojuegos. Su precio, la facilidad de manejo y la gran cantidad de herramientas que posee para el modelado poligonal, le hacen ser la ayuda ideal para cualquier grafista. En esta entrega modelaremos una nave aerodeslizadora con tan solo 48 polígonos. De esta manera, podremos ver que no hace falta recurrir a mallas densas para aplicar detalles a un objeto. La calidad de la textura será el factor determinante a la hora de controlar el realismo del objeto en cuestión.
Lo primero a realizar es un boceto de la idea que tenemos en mente. La nave deslizadora es de carácter bélico y de ambientación futurista. La línea típica para esta época es aerodinámica. Se ha realizado un dibujo a partir de estas ideas y se ha dividido al vehículo en varias partes:



FIGURA 1

- El cuerpo central. Esta parte de la nave albergará los generadores de energía y la maquinaria principal.
- La turbina. Se encargará de la propulsión principal. Sale por la parte superior del cuerpo y está situada en la parte trasera de la nave.
- Alerones traseros. Proporcionan propulsión secundaria y se encargan de hacer girar el vehículo.
- Las alas estabilizadoras laterales. Con ellas se consigue una alta maniobravilidad. Están situadas a cada lado de la nave.
- La cabina. Forma parte del cuerpo central y será donde se introducirán los ocupantes. Contiene todos los paneles y mandos de control.

Habiendo esbozado el dibujo, se comenzará a hacer la textura para la nave. Normalmente suele modelarse primero la malla para, posteriormente, aplicarle un bitmap. Nosotros lo haremos del otro modo, ya que contamos con una herramienta, dentro de las últimas versiones del MAX, muy útil para estos casos. Se trata del modificador llamado *UVW Unwrap*, con la que tendremos total control sobre la aplicación de la textura al objeto. Dada la posibilidad de aplicar partes de la imagen a determinadas zonas, se ha recurrido a este método. Ésta tendrá un tamaño final de 256x256 píxeles, un valor muy recomendado para objetos de este tipo en aplicaciones que utilicen estas aceleradoras gráficas.



FIGURA 3

Como puede observarse, la textura contiene todas las partes de la nave. Con ello tendremos un único fichero de imagen para todas las partes de la nave. De esta forma aprovecharemos enormemente la memoria de la aceleradora y se evitará tener que manejar multitud de ficheros por parte del programador.
Se ha recurrido al programa Adobe Photoshop para realizar la textura. Se han dibujado los alerones laterales en la parte izquierda, seguidos de un lado del cuerpo principal y el propio cuerpo. Más a la derecha se encuentran las diferentes partes que componen los

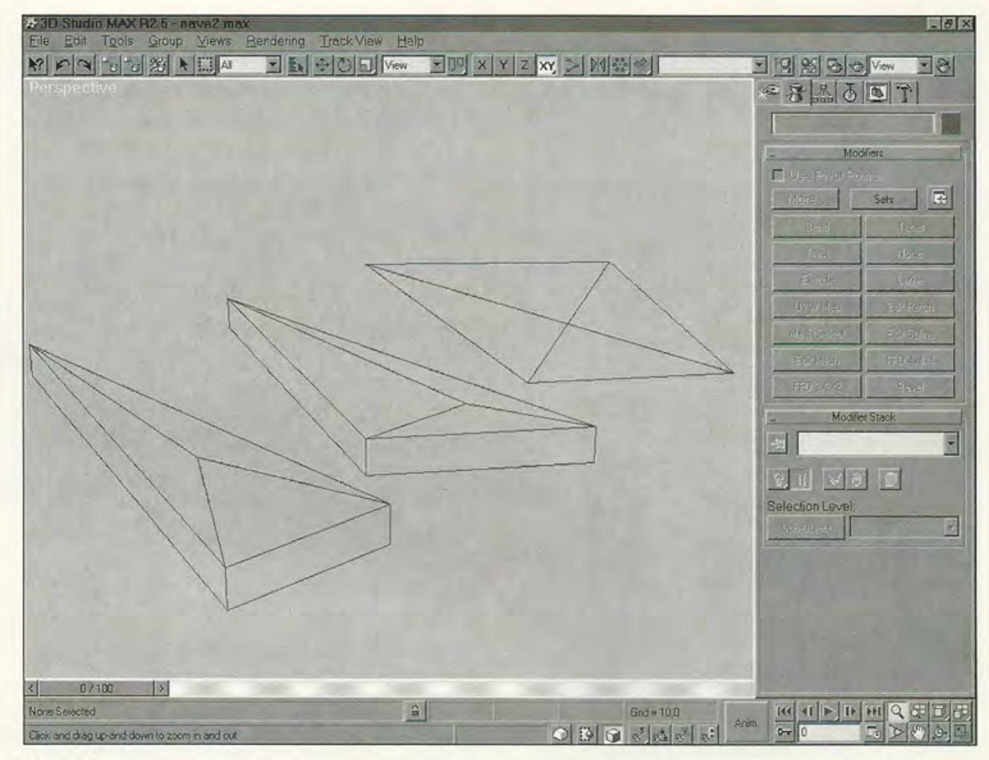


FIGURA 4

alerones traseros y la turbina. En la parte superior de la textura se ha dibujado un rectángulo que se aplicará a partes poco visibles del objeto, como pueden ser las esquinas de los alerones traseros, o la parte inferior del vehículo.

Se ha recurrido al programa Adobe Photoshop para realizar la textura

Pasamos a la parte del modelado de la nave. Como hemos comentado anteriormente, 3D Studio Max tiene multitud de herramientas y modificadores para trabajar con objetos poligonales. Así, es muy fácil modelar un objeto complejo basándonos en primitivas. Esto es lo que haremos para modelar parte de la nave. Comenzaremos el cuerpo central. Para ello crearemos un triángulo desde el menú *Shapes* haciendo uso de la opción line. Esta figura será la parte lateral. Si extruímos este triángulo obtendremos ya el objeto entero (figura 3). Para darle la forma deseada pasaremos a aplicarle un *Edit Mesh*. Una vez dentro de este modificador pulsamos sobre el botón *sub-object* para editar directamente los vértices. Moveremos ligeramente los dos superiores hacia afuera. Puesto que la nave diseñada tiene una cabina que se va estrechando, moveremos también los dos posteriores ligeramente hacia adentro. Al objeto le pondremos el nombre de "Cuerpo Central". Como hemos visto, ya tenemos la cabina y el cuerpo, formados por tan solo 8 polígonos. Pasamos ahora a las alas estabilizadoras. Desde el menú de creación de primitivas, seleccionaremos el prisma. Pulsando y arrastrando el ratón en la ventana *Top* obtendremos un mejor control de salida. En la figura 2 podemos ver los pasos a seguir para darle la forma adecuada.

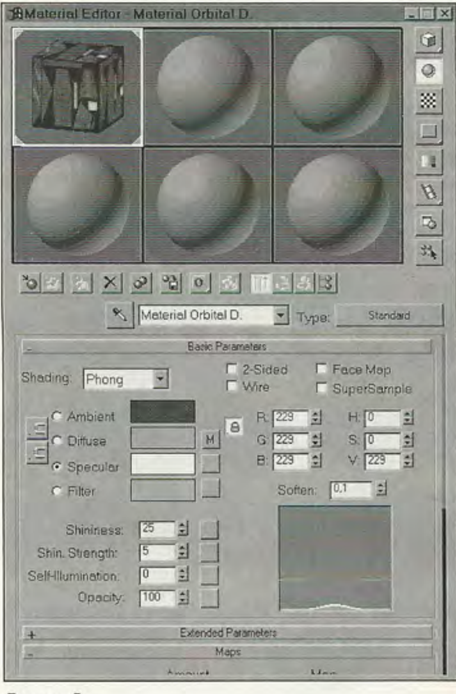


FIGURA 5

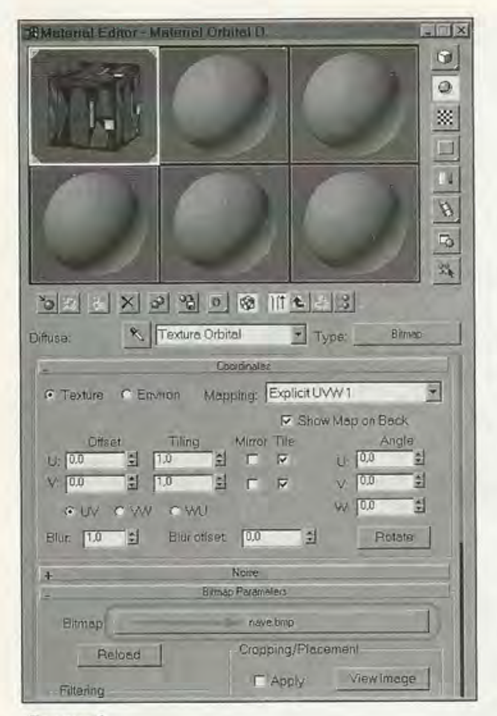


FIGURA 6

Una vez creado, le aplicamos un modificador *Edit Mesh* y pasamos a editar sus vértices. Seleccionamos los vértices de la esquina inferior derecha y los movemos hacia atrás. Los dos centrales los moveremos de tal forma que las rectas formadas entre los vértices superiores e inferiores y los centrales e izquierdos queden perpendiculares, como muestra la figura 4. Una vez hecho esto pasamos a editar las caras del prisma. Activamos la opción para seleccionar caras cuadrangulares, que es un botón con un dibujo de un cuadrado.

Los alerones traseros se crearán a partir de figuras utilizando rectas desde el menú Shapes

Arrastrando el ratón seleccionamos todas las caras menos las superiores y tras esto las borramos, obteniendo un objeto formado por 3 polígonos. Para comenzar a darle la forma adecuada, moveremos el vértice central hacia arriba. Al haber borrado las caras anteriores tendremos que crear una de forma que el objeto tenga una base inferior. Para ello usaremos la opción *build face* dentro del menú de edición de caras. El programa necesitará que le indiquemos 3 vértices para crear una cara triangular. Elegiremos los que rodean al central. Con ello conseguiremos un estabilizador lateral con tal sólo 4 polígonos! Para tener un suavizado correcto entre aristas, es necesario pulsar sobre el botón *auto-smooth* dentro del menú de edición de caras, habiendo elegido previamente todas las que forman al objeto. A éste lo llamaremos *"Ala estabilizadora izqda."*. Dado que el vehículo es totalmente simétrico, habiendo dibujado una ala podemos duplicar ésta. Para ello haremos uso de mirror con la opción *copy* activada. El eje de simetría será el de las X. Pasaremos a denominar este objeto *"Ala estabilizadora dcha."*. Ahora modelaremos la turbina. Este objeto se creará de forma idéntica al cuerpo central, es decir, a partir de una extrusión de un triángulo. Tenemos que tener en cuenta que la base de esta figura tendrá que ser paralela con la arista del cuerpo central donde va a ser colocada. Una vez modelada, la pondremos en su lugar correspondiente: la parte anterior del vehículo.

Los alerones traseros se crearán a partir de figuras utilizando rectas desde el menú *Shapes*. Son polígonos cuadrangulares con el mismo caso que el objeto anterior; la base del alerón la haremos también paralela a la arista del cuerpo central donde va a ser encajada la pieza. Una vez hecho esto, se extruirá el polígono. Para una mayor sensación de dinamismo, rotaremos el objeto en el eje Z desde el menú *Front*. A este objeto le llamaremos *"Alerón trasero izqdo."*. Al igual que la ala estabilizadora, haremos una copia de éste con la opción mirror. Colocaremos el alerón nuevo en su sitio y lo llamaremos *"Alerón trasero dcho."*.

Ya hemos terminado la fase de modelado del vehículo en cuestión. Si hacemos clic sobre la opción *more* en el panel de herramientas y pinchamos sobre *Polygon Count* podremos ver el número de polígonos de cada objeto y de la escena completa. El resultado final es el siguiente:

- Cuerpo Central: 8 polígonos.
- Turbina: 8 polígonos.
- Ala estabilizadora izqda: 4 polígonos.
- Ala estabilizadora dcha: 4 polígonos.
- Alerón trasero izqdo: 12 polígonos.
- Alerón trasero dcho: 12 polígonos.
- Total: 48 polígonos.

Para terminar formando un objeto único, desde el modificador Edit Mesh del cuerpo central, pulsamos sobre la opción *Attach* e iremos seleccionando todos los objetos creados. Finalmente, a este último le llamaremos "Orbital Destructor".

Ahora comenzaremos con el texturizado. Desde el Editor de Materiales pasamos a crear uno nuevo llamado "Material Orbital D.". Le daremos los siguientes valores:

- Ambient: R:25 G:25 B:25
- Diffuse: R:128 G:128 B:128
- Specular: R:229 G:229 B:229

Para incluir la imagen creada desde Photoshop pulsamos sobre el botón *Diffuse* en el panel de mapas. Seleccionamos como tipo de mapa un bitmap. Una vez hecho esto, cogemos la textura (en nuestro caso nave.bmp). Si pulsamos sobre el botón Aplicar Material podremos ver en las ventanas de previsualización con sombreado que el render se hace usando los valores Ambient, Diffuse y Specular. Para poder ver en tiempo real la textura aplicada, pulsamos, dentro del panel del bitmap, sobre el icono de la esfera con cuadrados azules y blancos.

Una vez terminada la aplicación del material, pasamos a mapear la textura sobre la malla. Para ello seleccionamos el Orbital Destructor y nos dirigimos a editar las caras. Pulsando el botón del cuadrado, al igual que antes, tendremos activada la opción para seleccionar caras cuadrangulares. Comenzaremos con el cuerpo principal. Desde la ventana Top se hará de manera cómoda. Seleccionaremos las 2 caras cuadrangulares superiores del cuerpo, es decir, la que será la cabina y la parte delantera del vehículo. Una vez hecho esto le aplicamos un modificador "UVW Map".

Para tener una perfecta organización de las modificaciones y nombres de los objetos

El tipo de mapeado que elegiremos será de tipo planar y se alineará de forma que quede paralela a la base del deslizador. Ahora pasaremos a la fase más importante de la aplicación de la textura al objeto. Pinchando sobre el botón de *more* dentro del panel de modificadores, encontraremos uno llamado *UVW Unwrap*, el cual seleccionaremos. Dentro de los parámetros de UVW Unwrap se encuentra un botón llamado *Edit*. Si lo pulsamos nos aparecerá en pantalla una ventana con los vértices y aristas de las caras que hemos seleccionado (en este caso las del cuerpo). Utilizaremos como imagen de fondo *nave.bmp*. Vemos cómo el programa nos coloca en la ventana el bitmap que vamos a utilizar como textura. Moviendo los vértices de estas caras a sus puntos respectivos en la imagen, obtendremos una perfecta aplicación de la textura sobre el objeto. Para tener una perfecta organización de las modificaciones y nombres de los objetos, pinchamos sobre el botón *Edit Stack*. De esta forma podremos ver todo el historial de creación del objeto. Cambiamos el nombre de "Edit Mesh" a "Cuerpo Superior". Nos ponemos en UVW Unwrap y pulsamos sobre *Reference Object*. De esta forma se creará una línea divisoria, pudiendo distinguir mejor las modificaciones que van sufriendo las diferentes partes de la nave. Cerramos la ventana y pasamos a aplicar al objeto un *Mesh Select*. Este modificador es



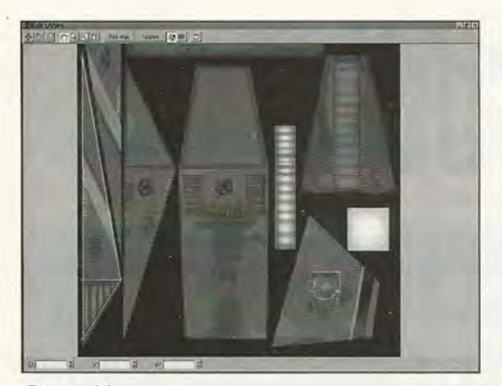


FIGURA 11

similar a la edición de caras de Edit Mesh. Seleccionaremos ahora la cara triangular del lado izquierdo del cuerpo del deslizador. Le aplicamos un mapeado planar, alineado con respecto a su cara. Para controlar la aplicación de la imagen a esta parte del vehículo pulsamos sobre *UVW Unwrap*. De nuevo seleccionamos *nave.bmp* como fondo de referencia. En la ventana nos habrá aparecido un triángulo. Moveremos el vértice correspondiente a la parte frontal del lado de la nave a la parte superior de la textura, de forma que concuerde con su sitio. Los dos vértices restantes irán a la parte inferior de la imagen, donde termina la zona correspondiente al lateral de la nave. De nuevo editaremos el historial del objeto, renombrando "Mesh Select" a "Lateral Izqdo Cuerpo". Para no confundirlo con las siguientes ediciones, pulsaremos sobre *Reference Object*, que creará una nueva línea de separación. Dado que el otro lado de la nave es simétrico con respecto a éste, podremos utilizar para él la misma zona de la textura. Tras haber hecho las modificaciones, llamaremos a esta parte "Lateral Dcho Cuerpo" e insertamos otra línea de división. El que haya simetrías en la textura viene muy bien, ya que se utiliza para varias caras la misma zona del bitmap, pudiendo dejar el resto de la imagen para detallar con más énfasis otras partes del objeto.

Para la parte inferior del cuerpo central utilizaremos el rectángulo superior de la textura. La escala y las proporciones no importan, ya que el programa se encarga automáticamente de estirarla para adecuarse al tamaño de éstas. Dado que la parte inferior de la nave no va a verse demasiado, se ha optado por hacer este rectángulo de un tamaño no demasiado grande. Como dije antes, aprovecharemos esto para que los dibujos de otras partes de la nave sean más amplios, pudiéndose aplicar más detalles. Una vez terminado con el cuerpo central, pasaremos a mapear la turbina. Aplicando otro *Mesh Select* seleccionaremos todas las caras, menos la inferior y la trasera, que componen esta parte de la nave. Se aplicará un UVW Map y posteriormente un UVW Unwrap. Los dos vértices posteriores corresponden a la parte más adelantada de la turbina. Debe dejarse espacio entre las rejillas de ventilación y las aristas que determinan la parte superior de la turbina. Colocar el resto de los vértices en sus respectivos sitios es tarea fácil. Para la parte trasera del objeto se usará el cuadrado azul de la textura, que simula un fuego de este color. A la base se le puede aplicar perfectamente, al igual que la parte inferior de la nave, el rectángulo gris.

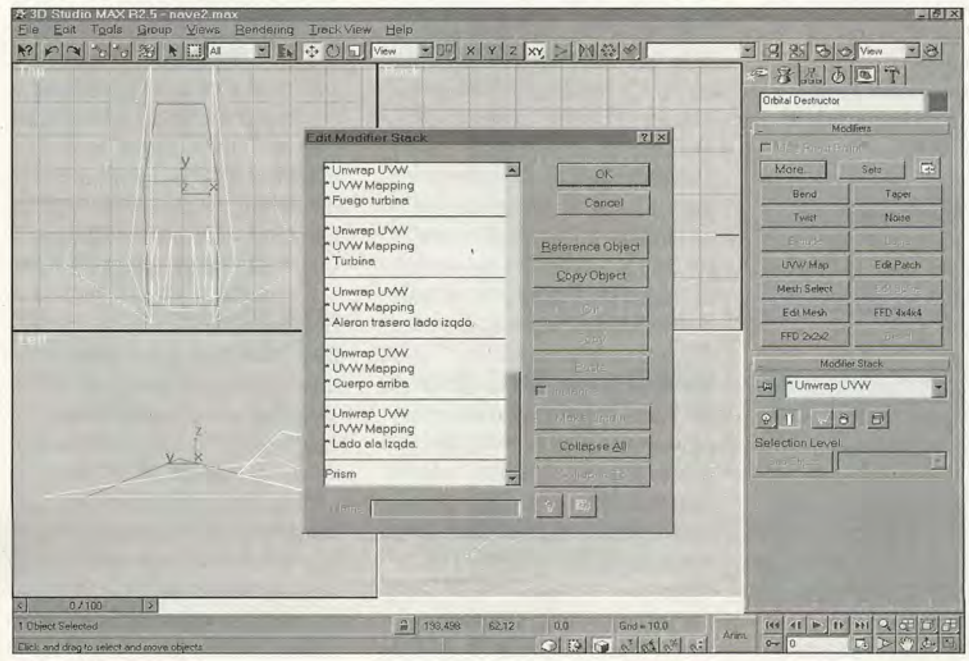


FIGURA 10

Pasamos con los alerones traseros. Tan solo hemos creado en la textura un lateral del alerón que servirá para ambos lados, con la salvedad de que hay que tener precaución con la simetría. La imagen lleva un pequeño logotipo que tiene que estar siempre orientado hacia el mismo lado. Si no tenemos esto en cuenta, tendremos siempre un alerón con la textura invertida. Para evitar errores, la aplicaremos primero a un lado y luego al otro. La parte trasera del elemento llevará por textura el rectángulo azul. Los alerones ofrecen una impulsión secundaria a la nave y pueden girar para rotarla. A las caras restantes se aplicará el rectángulo grisáceo.

Aunque dependiendo de la movilidad del vehículo en el juego habría que hacer modificaciones

Para la ala estabilizadora seleccionaremos primero las dos caras laterales, aplicándoles la figura en el extremo izquierdo del bitmap. Tendremos en la ventana de UVW Unwrap una figura con cuatro vértices. El vértice correspondiente a la punta de la ala irá en el extremo superior de la figura, mientras que el correspondiente a la parte trasera irá al extremo inferior. Los dos restantes tendrán que colocarse en los extremos laterales, dejando la línea gris por debajo de ellas. A las caras restantes le aplicaremos de nuevo el rectángulo gris.

La forma en que será mapeado la ala derecha no llevará el mismo proceso. Dado que la figura lleva unas pequeñas letras inscritas, no se podrá aplicar a estas caras, ya que la inscripción aparecería invertida. Para ello aplicaremos primero la 2ª figura de la textura a la cara exterior del ala. Posteriormente aplicaremos la rejilla a la parte anterior del elemento y el rectángulo gris para las restantes.

De esta forma habremos completado el modelado y texturización del Orbital Destructor. Aunque dependiendo de la movilidad del vehículo en el juego habría que hacer modificaciones, como separar la nave por objetos, serían de escasa importancia y muy sencillas de realizar.

En este tutorial se ha enseñado a utilizar el modificador UVW Unwrap, de vital importancia para la texturización y mapeado en baja poligonización. Con él se consigue una perfecto control sobre la aplicación de la textura en objetos. Como podemos ver, no hacen falta excesivos polígonos para conseguir el efecto deseado. Sabiendo todos estos detalles esperamos que pronto tengáis total facilidad para crear vuestros objetos para incluir en juegos tridimensionales. ¡Hasta otra!

Autor: Pablo Trinidad

Dlls de DIV Games Studio

Las Dlls son, tal vez, uno de los puntos fuertes del entorno de DIV. Es conveniente conocerlas a fondo a medida que nos vayamos internando en un mundo que nos va a abrir de forma sencilla las puertas de la programación de videojuegos.

Los programas de DIV tienen un punto fuerte y es la utilización de una librería externa o Dll, donde se concentran todas y cada una de las funciones del lenguaje DIV. Esta librería es *div32run.dll*. Con esto se evita la inclusión de todas aquellas funciones innecesarias en el código del ejecutable de nuestro juego, cargando en memoria sólo aquellas que son estrictamente necesarias. Con ello, el archivo ejecutable se convierte en un simple cúmulo de datos del juego y llamada a las funciones presentes en la librería *div32run.dll*.
Esto no parece ser relevante, pero esta posibilidad de llamar a funciones externas permite al lenguaje DIV no sólo acudir a la dll principal, sino a otras dlls de construcción propia o no. Con esto podemos ampliar el lenguaje DIV y dotar de nuevas capacidades a éste.
Aprenderemos cómo podemos utilizar las Dlls, las herramientas necesarias para construirlas, así como los distintos tipos. Con las dll's podemos ampliar el lenguaje DIV y dotarlo de nuevas capacidades

TIPOS DE DLLS

Podemos distinguir tres tipos de Dlls:

- **Autocarga:** son aquellas que pueden sustituir algunas de las funciones internas de DIV. Se denominan así porque se acoplan inmediatamente a cualquier juego sin necesidad de cambiar el código.

Tenemos que conocer las librerías que nos ofrece DIV para poder sacarle todo el partido

- **Salvapantallas:** al igual que los salvapantallas de Windows, son pequeños programas que transcurrido un tiempo de espera muestran un efecto en la pantalla para volver a la situación anterior una vez pulsada una tecla, movido el ratón o joystick.
- **De Funciones:** añade nuevas funciones al lenguaje DIV.

CÓMO UTILIZAR DLLS

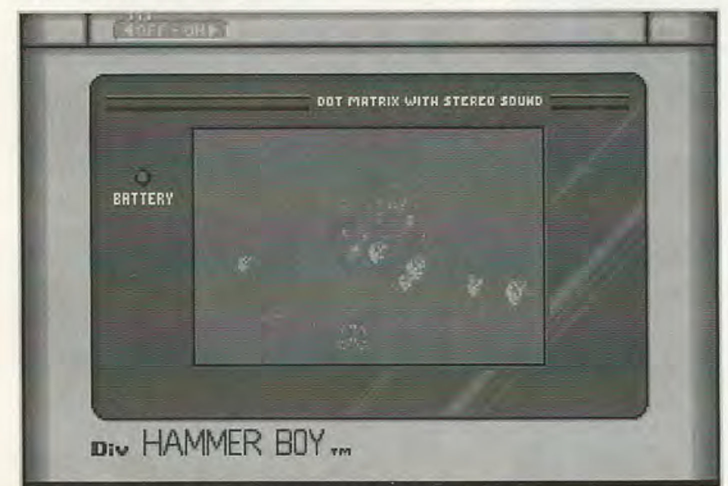
La utilización de Dlls no es complicada. Si la librería a utilizar es una Dll de autocarga o un salvapantallas, basta con añadir el fichero en cuestión al directorio donde se encuentra el ejecutable que va a utilizar dicha Dll. Si queremos que también funcionen en todos los juegos que ejecutemos en el entorno de DIV, tan sólo tendremos que añadir la Dll al directorio donde tenemos instalado DIV, es decir, donde se encuentra el ejecutable principal d.exe.
Para utilizar las Dlls de funciones, debemos cargarlas manualmente. Para ello, sólo tenemos que añadir la siguiente sentencia tras las declaraciones LOCAL del programa principal:
IMPORT "directorio\nombre.dll"
El directorio, por defecto, es aquel donde se encuentra el ejecutable del juego. Una vez hecho esto, todas las funciones que incorpora la librería estarán disponibles para el juego.

CONSTRUCCIÓN DE DLLS

Así como la utilización de estos módulos es bien sencilla, su creación no lo es tanto. Para ello necesita de un profundo conocimiento del lenguaje C, así como de *buffers* y manejo de datos.
Las Dlls que utiliza DIV Games Studio son bibliotecas de enlace dinámico de Windows NT, por lo que es necesario tener un compilador de C que genere dicho tipo de archivo. El utilizado por el equipo de desarrollo de DIV es el Watcom C++ Compiler, aunque también puede compilarse con Borland C++ 4.X o superior. Para más información acerca de la utilización de este último compilador, visite la sección *downloads* de la página web oficial de DIV Games Studio (*www.divgames.com*).
El porqué del uso de estas Dlls es sencillo. Hacen uso de la tecnología de 32 bits, al igual que el entorno de DIV, aprovechando al máximo las capacidades del compilador. Además es un formato de alguna forma accesible a usuarios de nivel avanzado, que son los que se presuponen van a fabricar este tipo de módulos.
En este número vamos a aprender los principios básicos del desarrollo de Dlls de funciones. DIV Games Studio utiliza dll's de Windows NT.

DLLS DE FUNCIONES

En primer lugar, para poder utilizar todas las funciones de desarrollo de dlls debemos cargar el archivo de cabecera *div.h*, además de definir una constante de preprocesador llamada GLOBALS. Estas operaciones se realizarán después de cargar los archivos de cabecera necesarios para el módulo en cuestión. Lo vemos con un ejemplo:



PODRAS SENTIR LAS SENSACIONES DE LA CLASICA GAME BOY.

```
#include <stdio.h>
#include <conio.h>

#define GLOBALS
#include "div.h"
```

Una vez realizado esto, estamos en disposición de utilizar todas las opciones que nos ofrece DIV. En primer lugar debemos implementar dos funciones imprescindibles en cualquier módulo de este tipo. Su estructura es la siguiente:

```
void _export
divlibrary(LIBRARY_PARAMS) {
//Aquí irán las declaraciones de
las funciones a exportar.
}

void _export
divmain(COMMON_PARAMS) {
  GLOBAL_IMPORT();
}
```

Tras esto, ya estamos en disposición de crear nuestra primera función para DIV. Vamos a elaborar una función que calcule el seno de un ángulo, bastante útil a la hora de realizar algunos efectos en demos. Hay que tener en cuenta que el valor que recibiremos indicará el ángulo en coma fija con tres decimales, es decir, 45,345 grados serán 45345 grados. Crearemos una función llamada *seno()*. Recibirá un parámetro que indicará los grados del ángulo en cuestión. A la hora de crear Dlls, los parámetros no se escriben entre paréntesis sino que deben asignarse a variables en la propia implementación con la función *getparm()*. Asimismo la función debe declara como *void* el tipo de dato a devolver. Lo vemos mejor con el código de la función:

```
void seno()
{
int angulo=getparm();
}
```

Todas las funciones deben devolver un valor aún si no es necesario. Para ello, utilizaremos la función *retval(int)*, para indicar el valor a devolver. En nuestro ejemplo, debemos convertir el ángulo dividiendo el parámetro recibido entre 1000 para, posteriormente, realizar la operación inversa con el resultado, tomando los tres dígitos más significativos. Para simplificar, tan sólo tomaremos los grados sin decimales, por lo que no usaremos *floats*. La función completa sería:

```
void seno()
{
int angulo=getparm();
int
resultado=int(sin(angulo/1000)*1
000);
retval(resultado);
}
```

Una vez realizada la implementación de nuestra función, solamente debemos exportarla a DIV. Para ello utilizaremos la función *COM_export()*, que recibe tres parámetros:

- el primero indica el nombre que la función va a tener en DIV
- el segundo es el nombre de la función asociada, en nuestro caso *seno()*
- zel último indica el número de parámetros que recibe dicha función.

En el ejemplo sería:

```
COM_export("SENO",seno,1);
```

Esta operación se realiza dentro de la función *divlibrary()*. Debemos recordar que hemos de llamar a los archivos de cabecera necesarios, en este caso *math.h* y *stdio.h* (ya que lo utiliza *div.h*). Por tanto, nuestro código final será:

```
#include <stdio.h>
#include <math.h>

#define GLOBALS
#include "div.h"

void seno()
{
int angulo=getparm();
int
resultado=int(sin(angulo/1000)*1
000);
retval(resultado);
}

void _export
divlibrary(LIBRARY_PARAMS) {
COM_export("SENO",seno,1);
}

void _export
divmain(COMMON_PARAMS) {
  GLOBAL_IMPORT();
}
```



SE TRATA DE UN EFECTO SENCILLO PERO MUY ATRACTIVO.

Guardamos el archivo de código con el nombre que deseemos, por ejemplo *seno.cpp*. Ya puede compilar la Dll y corregir los posibles fallos de compilación que le hayan surgido.

La creación de las DLLs es bastante complicada, pero su uso es muy sencillo

Se dará cuenta de que hay un fallo en la declaración de la función *put_sprite* en *div.h*. Debemos sustituir la palabra *byte** por *unsigned char*. Con esto solucionaremos el problema y podremos continuar con la compilación.
Esto es todo por este número. Practique con funciones sencillas de este tipo y pruébelas con programas de DIV. Asimismo, puede utilizar como referencia el ejemplo *demo2.dll* incluido en el directorio dll de DIV. En nuestro próximo número profundizaremos más en la creación de Dlls.

LOS EJEMPLOS DEL DIV

DIV Games Studio incluye unos ejemplos de la construcción de Dlls que pueden servirnos para entender la utilidad de estos módulos. Estos son:
ss1.dll: salvapantallas que simula un fundido en forma de granos de arena
agua.dll: realiza un efecto de agua en la zona inferior de la pantalla. Basta con incluirlo en el directorio de nuestro juego y... ¡ya está!
hboy.dll: simula a la famosa Game Boy, ejecutando el juego en cuestión emulando dicha consola. Un ejemplo que merece la pena comprobar su funcionamiento. También es una dll de autocarga.
demo2.dll: añade la capacidad de calcular raíces cuadradas a cualquier programa.
Así también podemos encontrar varios ejemplos de Dlls creadas por usuarios de DIV y cedidas como *freeware* en la página web oficial de DIV Games Studio, como por ejemplo *polys.dll* (dibuja líneas, cuadrados y círculos), *ascii.dll* (manejo de datos ASCII y entrada de datos por teclado) y *tad.dll* (manipula pilas y colas).
Hemos realizado una pequeña introducción al sistema de Dlls de DIV. En el próximo número continuaremos con la construcción de estos módulos tan importantes en nuestro entorno.

Autor: Carlos Glez. Morcillo

Golgotha... más que un simple código fuente

Tras un par de meses de descanso en esta sección, volvemos con una verdadera joya de código fuente que incluye modelos 3D, música, gráficos y herramientas de desarrollo totalmente gratuitos... Estamos hablando del proyecto Golgotha de la empresa, recientemente desaparecida, Crack dot Com. ¿Alguien da más?

Crack.Com, la famosa empresa desarrolladora de *Abuse* (juego que les ingresó más de 33 millones de pesetas simplemente por royaltys), cerró por problemas económicos el año pasado (una mala gestión en sus asuntos financieros). Cuando los chicos de Crack Com acabaron *Abuse*, se propusieron comenzar un nuevo proyecto. Barajaron varias posibilidades, como hacer un juego tipo *Gauntlet* en 3D, o uno tipo *Command and Conquer*, tal vez un juego tipo *Doom*... Mezclando la acción de *Doom* y la estrategia de C&C nació Golgotha.

El código fuente de Golgotha es totalmente gratuito, podéis modificarlo, incluir partes suyas en vuestros propios códigos...

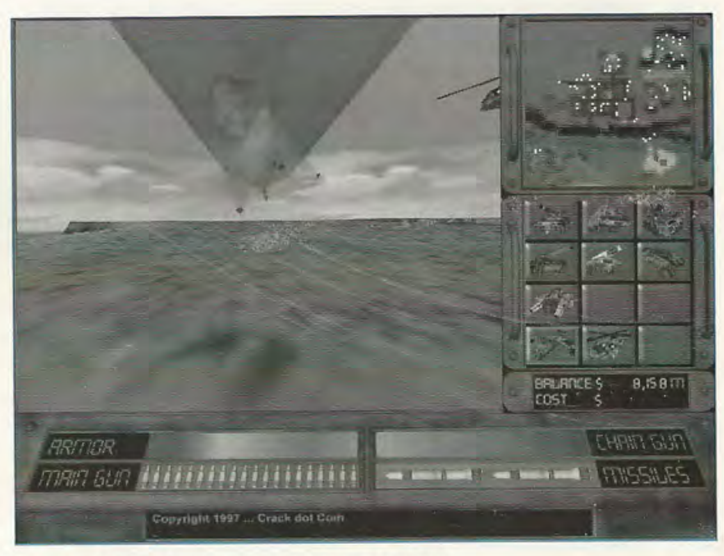
Allá por 1995 comenzó el desarrollo de este proyecto; un tremendo videojuego 3D que funcionaría en Windows 95 y Linux. Por diversos problemas, este ambicioso proyecto quedó inacabado (pero en fase de desarrollo bastante avanzada), y los propietarios de la empresa Crack dot Com (Dave Taylor y Jonathan Clark) decidieron poner el código fuente y los ficheros de gráficos, música y modelos del videojuego a disposición de todos (y con unas condiciones de utilización únicas).



El código fuente de Golgotha es totalmente gratuito, podéis modificarlo, incluir partes suyas en vuestros propios códigos... todo sin ningún tipo de restricción. Hay que exceptuar las partes que no son propiedad de Crack dot Com, como el código que carga y la grabación en formato jpeg, el lector de MP3 (que tiene una patente bastante severa), además de los logotipos de marcas registradas como Golgotha y Crack dot Com. Exceptuando esto, el resto es de dominio público, y podéis utilizarlo como queráis.

CARACTERÍSTICAS DE GOLGOTHA

El código fuente de Golgotha incluye un completísimo motor 3D con opciones de luces, aplicación de texturas, filtros... (que puede ser usado para gran variedad de videojuegos), soporte total para Glide y parcial para Direct3D y OpenGl. Pasando al sonido, soporta DirectSound, Direct3dsound, Aureal 3D sound y parcialmente soporta sonido para Linux. Se incluyen fuentes para cargar Wav y MP3, además de herramientas para trabajar con los ficheros de sonido del juego. Los modelos están realizados en 3D Studio MAX, y con un plug-in "de cosecha propia", se pasan a un formato directamente utilizable por el motor de Golgotha. Para aplicar texturas y editar los modelos 3D realizaron sus propias herramientas (que también incluyen). Además de todo esto, el programa soporta juego en red, tiene una interfaz propia de ventanas, botones, etc. (no utiliza los de Windows), está optimizado para Pentium y k6, etc.



Efectos sorprendentes para el alcance de todos.

En definitiva, más de 5 Mb de código fuente (sí, ¡¡5 megas!!) sin desperdicio para todo programador que quiera meterse de lleno en programación 3D en Windows o Linux. Todo esto unido a las herramientas, música, gráficos y modelos totalmente gratuitos hacen de este paquete un documento técnico imprescindible para todo grupo de desarrollo. Para obtener más información del videojuego, enviar un e-mail con vuestras dudas a los programadores o grafistas que intervinieron en su desarrollo, o informaos de los grupos que se están formando para terminar Golgotha, os podéis pasar por la página oficial del proyecto en *http://www.crack.com/golgotha_release/*



Autor: José Antonio Guerra

Introducción al sonido posicional con DirectSound

En estos tiempos de predominio de los videojuegos tridimensionales es obligado hacernos un hueco en esta sección y hablar del sonido 3D, elemento fundamental para dotar de realismo y espectacularidad a estas aplicaciones. Veamos los conceptos fundamentales de esta técnica.

Este mes vamos a ver una introducción al sonido posicional, sus características básicas y los métodos que DirectSound nos facilita para implementarlo en nuestras aplicaciones.

¿QUÉ ES EL SONIDO POSICIONAL?

El sonido posicional dota a nuestros efectos sonoros de posición en el espacio y de ciertas características físicas. Su misión fundamental es de complemento para entornos tridimensionales, ya que no sólo de gráficos vive el hombre. Los sonidos originales son procesados según su distancia al oyente, su posición relativa y su velocidad. Por ejemplo, si un emisor de sonido - un altavoz, sin ir más lejos – está situado a la izquierda del oyente, éste recibirá una señal más fuerte por su oído izquierdo, creando la sensación de posición. También el sonido es atenuado por la distancia, ya que un objeto se escucha cada vez más bajo conforme aumenta su distancia. Y la velocidad es también un factor muy importante, ya que genera el efecto *Doppler*, que muchos recordareis de vuestras clases de Física. Recordemos que este efecto se produce cuando un objeto se mueve con velocidad relativa respecto de un punto de referencia. Para un emisor de sonido, cuando dicho objeto se acerca, la frecuencia del sonido se hace más aguda, y cuando se aleja el sonido lo percibimos más grave. Este es un efecto físico muy común, sólo tenéis que fijaros cuando una ambulancia pase por la calle con la sirena puesta, por ejemplo.

En un entorno tridimensional también se pueden simular muchos más efectos, pero sólo los descritos anteriormente están contemplados en DirectSound. En un entorno realista, el sonido se ve afectado por las paredes y los objetos de la escena, que provocan reflexiones y ecos, potenciando de esa manera el realismo y la inmersión total del usuario.

DIRECTSOUND 3D

Vemos con más detalle las características de sonido posicional que contemplan las librerías:

PERCEPCIÓN DE POSICIONES

Los factores que DirectSound tiene en cuenta para la percepción espacial del sonido son las siguientes:

- **Volumen:** Se tiene en cuenta la atenuación por distancia. Cuanto más lejos está un objeto, más leve es su sonido.
- **Diferencia de llegada:** Cuando un sonido se produce a la izquierda del oyente, será percibido un poco antes por el oído izquierdo que por el derecho. Esta diferencia es de un milisegundo, aproximadamente.

Ya sabemos la máxima de las tres dimensiones: "El triángulo más rápiso es el que no se pinta"

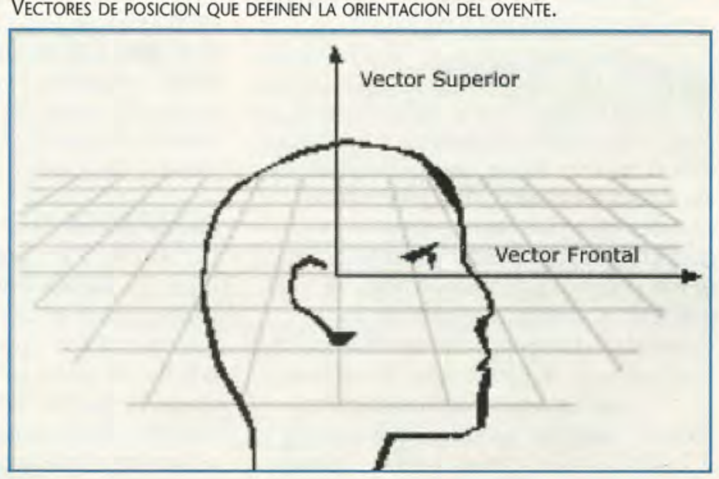
Una de las características más importantes del sonido posicional es la posición visual aparente de los emisores de sonido. Si un proyectil aparece como un punto lejano en la distancia y cuando se acerca se aprecia que es un misil intercontinental, cuando pase rugiendo y se aleje de nosotros por detrás, no necesitamos pequeños detalles en el sonido para que percibamos por dónde se ha marchado. Como podréis imaginar, éste es un factor muy importante en la optimización de nuestros programas, ya que el procesamiento de sonido sin un hardware dedicado es algo costoso. Y ya sabemos la máxima de las tres dimensiones: "El triángulo más rápido es el que no se pinta".

OYENTES

Los oyentes (*listeners*, en inglés) perciben el mismo efecto cuando un sonido se mueve en un arco de 90 grados alrededor suyo que si gira la cabeza esos mismos 90 grados. Por tanto, es mucho menos costoso en términos de eficiencia cambiar la posición del oyente que la de todos los objetos de una escena. Para este fin, DirectSound posee una interfaz: **IDirectSound3DListener.**

La orientación del oyente está definida por un par de vectores que comparten un mismo

VECTORES DE POSICION QUE DEFINEN LA ORIENTACION DEL OYENTE.



origen. Uno tiene origen en el centro y se dirige hacia arriba (vector superior o *top vector*) y otro, que comparte el mismo origen y forma un ángulo recto orientado hacia delante, a través de la cara del oyente (vector frontal o *front vector*).

CONOS DE SONIDO

Los conos de sonido sirven para dotar de orientación a los efectos. Un sonido sin orientación es un sonido puntual; la atenuación por distancia es la misma independientemente de la orientación (Ver figura).

En DirectSound, los conos de sonido incluyen un cono interior y un cono exterior. Dentro del cono interior, el volumen está al máximo nivel para esa fuente de sonido. Fuera del cono exterior, el volumen es el especificado para el exterior más el volumen interior. Si una aplicación, por ejemplo, especifica el volumen exterior a *DSBVOLUME_MIN*, el sonido será inaudible fuera del cono exterior. Entre el cono interior y el exterior, el volumen cambia gradualmente de un nivel a otro.

Técnicamente, cada uno de los buffers de sonido representados por la interfaz **IDirectSound3DBuffer** es un cono de sonido, pero frecuentemente dichos conos actúan como sonidos puntuales. Por ejemplo, el valor por defecto para el volumen fuera del cono exterior es cero; mientras que la aplicación no cambie este valor, el volumen será el mismo tanto fuera como dentro de dicho cono, y el sonido no poseerá orientación aparente.

También se puede realizar cambiando el ángulo de los conos, que si es suficientemente grande será una esfera.

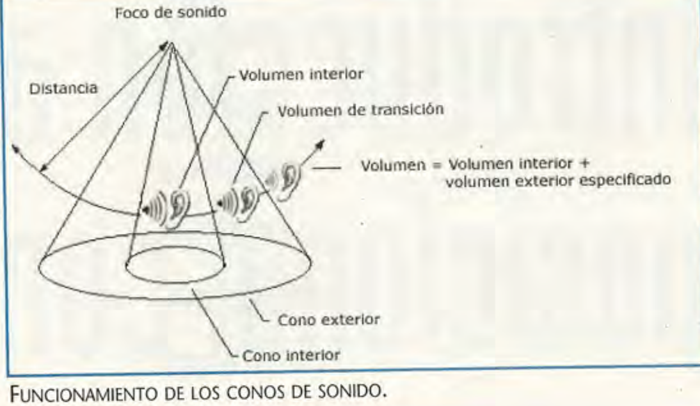
El diseño correcto de los conos de sonido es un elemento fundamental para crear ambientes realistas. Por ejemplo, podemos situar un sonido en el centro de una habitación y orientado hacia una puerta. Entonces establecemos el ángulo del cono exterior para que cubra todo el pasillo, e indicamos que el cono exterior sea inaudible. El usuario, cuando entre por la puerta, escuchará de repente el sonido emanando desde la habitación.

MEDIDAS DE DISTANCIA

Los efectos tridimensionales de DirectSound tienen como medida de distancia estándar los metros. Si vamos a crear una aplicación que no use metros, no es necesario convertir entre unidades de medida para mantener la compatibilidad con este componente. En lugar de eso, podemos especificar un factor de distancia, un valor en coma flotante que representa metros por unidad de distancia. Por ejemplo, si usamos pies en lugar de metros (un pie mide aproximadamente 30 centímetros), podemos especificar un factor de distancia de 0.30480006096, que es equivalente al número de metros en un pie.

Los valores de distancia usados para el sonido posicional imitan al mundo real. En el diseño de videojuegos, generalmente se tienden a exagerar estas medidas.

Mediante la exageración de efectos como el *Doppler* podemos ajustar ambientes sonoros que encajen perfectamente en nuestras ideas.



FUNCIONAMIENTO DE LOS CONOS DE SONIDO.

Conforme un oyente se acerque a una fuente de sonido, dicho sonido se escucha más alto. Aunque pasado un cierto punto, no es deseado que el volumen se incremente; o el nivel máximo ha sido alcanzado (0, ya que DirectSound no soporta amplificación de sonidos), o la naturaleza del sonido impone un límite lógico. Esta es la distancia mínima para la fuente de sonido. Por otro lado, la distancia máxima del sonido es aquella tras la cual el sonido es inaudible.

También hay que remarcar que si se crea un búffer 3D, no se pueden utilizar las funciones de balance de canales del búffer

La distancia mínima es especialmente útil cuando la aplicación debe compensar la diferencia entre volúmenes absolutos para distintos sonidos. Aunque el motor de un avión tiene una intensidad sonora mucho mayor que una abeja, por ejemplo, por razones prácticas dichos sonidos deben grabarse a unos volúmenes absolutos similares, ya que el sonido de 16-bits no tiene suficiente espacio para acomodar tales desfases de volumen. Podríamos usar una distancia mínima de 100 metros en caso de avión y de 2 metros en caso de la abeja. Con esos parámetros, el motor del avión se escucharía a la mitad de su intensidad a unos 200 metros del oyente, mientras que el volumen de la abeja sería la mitad tan sólo a 4 metros (Ver figura)

INTEGRACIÓN CON DIRECT3D

Las interfaces **IDirectSound3Dbuffer** y **IDirectSound3Dlistener** están especialmente diseñadas para ser utilizados conjuntamente con Direct3D. Los tipos **D3DVECTOR** y **D3DVALUE** usados en el sonido posicional usados también en dicho componente. Asimismo, DirectSound3D utiliza el mismo sistema de referencia de mano izquierda (ver figura). Todo esto permite la integración de gráficos y sonido mediante tecnología DirectX de una manera sencilla, aunque por supuesto pueda ser usada también en conjunción con otras APIss 3D, como Glide u OpenGL. En dichos casos, habrá que tener en cuenta el sistema de referencia utilizado en cada caso y realizar los ajustes precisos.

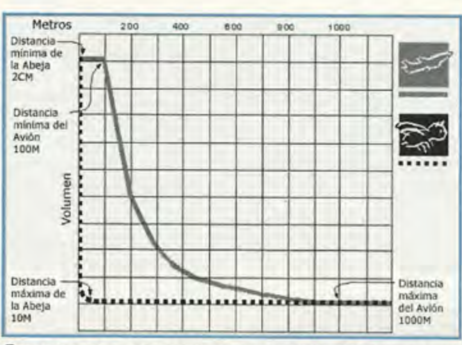
Efecto Doppler

DirectSound crea automáticamente el efecto Doppler para cada búffer u oyente que posea una velocidad. Dichos efectos son cumulativos: si el oyente y la fuente de sonido se están moviendo, el sistema automáticamente calcula la relación entre sus velocidades y ajusta el efecto Doppler en consecuencia.

La velocidad de una fuente de sonido o de un oyente no tiene por qué reflejar la velocidad real con la cual se está moviendo a través del espacio. La velocidad es simplemente un vector usado para calcular el efecto Doppler. Para crear un efecto Doppler realista, debemos calcular la velocidad para cada objeto en movimiento y establecer el valor correcto para cada sonido u oyente. Para crear efectos especiales, simplemente debemos exagerar o minimizar estos valores. Además, podemos incrementar o decrementar globalmente el efecto Doppler mediante el establecimiento de un factor Doppler para el oyente.

BUFFERS DE SONIDO 3-D

Los buffers de sonido 3D se crean y se administran igual que los buffers de sonido convencionales, tal y como vimos en anteriores entregas. La única particularidad es la necesidad de obtener una interfaz **IDirectSound3DBuffer** a partir del búffer original. El único requisito es que el *flag DSBCAPS_CTRL3D* sea especificado a la hora de la creación del búffer de sonido. También hay que remarcar que si se crea un búffer 3D, no se pueden utilizar las funciones de balance de canales del búffer. Si el búffer es creado con el *flag* DSBCAPS_CTRLPAN, nunca podrá funcionar como sonido posicional.



EFECTOS DE LOS VALORES DE DISTANCIA MINIMA Y MAXIMA DE UN SONIDO POSICIONAL.

Los buffers de sonido posicional no deberían almacenar sonidos estéreo. Debido a que el propio procesamiento del sonido posicional altera el canal en el cual se reproduce, es inútil usar sonidos con dos canales. De todas formas, si se diese este caso DirectSound convertiría el sonido estéreo en uno monoaural antes de efectuar ningún procesamiento, con el correspondiente gasto de procesador.

Obtención del interfaz IDirectSound3DBuffer

Para obtener un puntero a una interfaz **IDirectSound3DBuffer**, hemos de crear primero un búffer de sonido con el *flag DSBCAPS_CTRL3D*. Entonces, usaremos el método **IDirectSoundBuffer::QueryInterface** del búffer creado para obtener dicho puntero. Veamos un ejemplo de este método:

```
// LPDIRECTSOUNDBUFFER lpDsbSecondary;
// El buffer ha sido creado con el flag
DSBCAPS_CTRL3D.
LPDIRECTSOUND3DBUFFER lpDs3dBuffer;

HRESULT hr = lpDsbSecondary-
>QueryInterface(IID_IDirectSound3DBuffer,
                                &lpDs3dBuffer);
if (SUCCEEDED(hr))
  {
  // Establecemos los parametros 3D del buffer
  .
  .
  }
```

ESTABLECIENDO DISTANCIA MINIMA Y MAXIMA

La distancia mínima por defecto, *DS3D_DEFAULTMINDISTANCE*, está definida en dsound.h con el valor 1 (1 metro en situaciones normales). La distancia máxima por defecto, definida como *DS3D_DEFAULTMAXDISTANCE*, es infinito.

Para modificar o consultar el valor de mínima distancia podemos usar los métodos **IDirectSound3DBuffer::SetMinDistance** y **IDirectSound3DBuffer::GetMinDistance**, respectivamente. De la misma manera podemos gestionar la distancia máxima usando los métodos **IDirectSound3DBuffer::SetMaxDistance** y **IDirectSound3DBuffer::GetMaxDistance**.

ESTABLECIENDO EL MODO DE OPERACION

Los buffers de sonido posicional poseen tres modos de operación: normal, relativo al oyente (*head-relative*) y desactivado (*disabled*).

El ángulo de los conos por defecto es de 360 grados, lo que significa que a métodos prácticos son esferas

El modo normal es el valor por defecto. En el modo relativo al oyente, los parámetros de sonido (posición, velocidad y orientación) son relativos a los parámetros del mismo. En este modo, los parámetros del sonido son actualizados automáticamente cuando los parámetros del oyente cambian. En el modo desactivado, el procesamiento tridimensional es desactivado y los sonidos parecen ser emitidos desde el centro de la cabeza del oyente.

Para cambiar el modo de operación de un búffer de sonido posicional, se utiliza el método **IDirectSound3DBuffer::SetMode**, especificando dicho modo en el parámetro *dwMode* (*DS3DMODE_NORMAL, DS3DMODE_HEADRELATIVE* o *DS3DMODE_DISABLE*).

ESTABLECIENDO LA VELOCIDAD Y POSICION DEL BUFFER

Mediante los métodos **IDirectSound3DBuffer::SetPosition** e **IDirectSound3DBuffer::GetPosition** podemos consultar y cambiar la posición de una fuente de sonido en el espacio.

Para gestionar la velocidad de las fuentes de sonido en el espacio podemos usar los métodos **IDirectSound3DBuffer::SetVelocity** e **IDirectSound3DBuffer::GetVelocity**. Esta información será únicamente utilizada para calcular el efecto *Doppler* en los sonidos, y no modifican la posición de la fuente. Dicha velocidad está medida en unidades por segundo.

ESTABLECIENDO LOS CONOS DE SONIDO

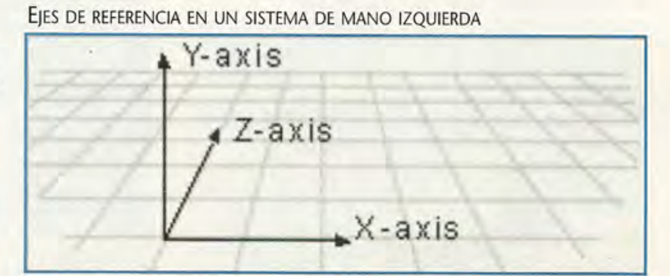
Para establecer y consultar los ángulos que definen los conos de sonido podemos utilizar los métodos **IDirectSound3DBuffer::SetCone Angles** e **IDirectSound3DBuffer::GetCone Angles**. Igualmente, con los métodos **IDirectSound3DBuffer::SetConeOrientation** e **IDirectSound3DBuffer::GetConeOrientation** podemos variar la orientación de dichos conos.

El ángulo de los conos por defecto es de 360 grados, lo que significa que a métodos prácticos son esferas – generan sonido en todas direcciones -. El cono exterior debe ser siempre igual o mayor que el cono inferior.

El cono exterior representa una atenuación adicional respecto del sonido original para cuando el oyente se sitúe fuera de los márgenes de dicho cono. Este factor está expresado en centésimas de decibelio, y su valor por defecto es cero, es decir, que no existe atenuación fuera del cono.

Para consultar y establecer el volumen de atenuación del cono exterior usamos los métodos **IDirectSound3DBuffer::SetCone OutsideVolume** e **IDirectSound3DBuffer::Get ConeOutsid eVolume**. Recordemos que el volumen del cono exterior está también sujeto a la atenuación por distancia.

EJES DE REFERENCIA EN UN SISTEMA DE MANO IZQUIERDA



Acumulación de parámetros

Los cambios en un búffer de sonido posicional, tales como posición, velocidad y factor Doppler originan que el mezclador de DirectSound remezcle el búffer de salida, consumiendo ciclos de CPU. Para minimizar este gasto de ciclos y mejorar el rendimiento, podemos ir acumulando cambios y efectuarlos todos juntos de una sola vez. Solamente tendremos que especificar el flag DS3D_DEFERRED en el parámetro dwApply de cualquier método de las interfaces IDirectSound3DListener o IDirectSound3Dbuffer que implique un cambio en la posición. Tras acumular todos los cambios que creamos convenientes, llamaremos al método IDirectSound3DListener::CommitDeferredSettings para efectuarlos todos de golpe, incrementando el rendimiento de nuestra aplicación, ya que sólo se efectuará una remezcla del búffer de salida.

Hay que remarcar que si especificamos un cambio por acumulación (con el parámetro DS3D_DEFERRED) y después un cambio inmediato (parámetro DS3D_IMMEDIATE) en la misma característica posicional, todos los parámetros acumulados serán sobrescritos por la última llamada. Por ejemplo, si establecemos la posición del oyente en el vector (1,2,3) con el flag DS3D_DEFERRED, después la establecemos a (3, 6, 9) con el flag DS3D_IMMEDIATE y después llamamos al método IDirectSound3DListener::CommitDeferredSettings, la posición del oyente será el vector (3, 6, 9)

OYENTES

Un oyente representa la persona que escucha los sonidos generados por un búffer de sonido. Está representado por la interfaz **IDirectSound3Dlistener**, que controla principalmente su posición y velocidad en el espacio. Además controla también la cantidad del efecto *Doppler* y de atenuación percibidas.

OBTENCIÓN DE LA INTERFAZ IDIRECTSOUND3DLISTENER

Para obtener un puntero a una interfaz **IDirectSound3Dlistener**, debemos crear en primer lugar un búffer primario de sonido posicional. Haremos esto llamando al método **IDirectSound::CreateSoundBuffer**, especificando los *flags DSBCAPS_CTRL3D* y *DSBCAPS_PRIMARYBUFFER* en el miembro *dwFlags* de la estructura **DSBUFFERDESC**. Entonces usaremos el método **IDirectSoundBuffer::QueryInterface** del búffer creado para obtener el puntero a la interfaz del oyente, como vemos en este ejemplo:

```
// LPDIRECTSOUNDBUFFER lpDsbPrimary;
// El buffer ha sido creado con el flag
DSBCAPS_CTRL3D.
LPDIRECTSOUND3DLISTENER lpDs3dListener;

HRESULT hr = lpDsbPrimary-
>QueryInterface(IID_IDirectSound3DListener,
                               &lpDs3dListener);

if (SUCCEEDED(hr))
  {
  // Seguimos nuestro camino...
  .
  .
  .
  }
```

ESTABLECIENDO EL FACTOR DE DISTANCIA

Como hemos comentado anteriormente, DirectSound utiliza como medida por defecto los metros, pero para usar cualquier otro tipo de medida proporcional podemos establecer un factor de distancia.
El factor de distancia se establece mediante el método **IDirectSound3DListener::Set DistanceFactor**. Una vez establecido, DirectSound se encargará automáticamente de convertir los parámetros usados a metros para sus cálculos internos.
Podemos consultar el factor de distancia actual invocando al método **IDirectSound3DL istener::GetDistanceFactor**. El valor por defecto es *DS3D_DEFAULTDISTANCEFACTOR*, definido como 1.0, lo que significa que cada unidad equivale a un metro. Con este factor,

Estableciendo el factor de atenuación por distancia

Podemos establecer y consultar el factor de atenuación por distancia mediante los métodos IDirectSound3DListener::SetRolloffFactor e IDirectSound3DListener::GetRolloffFactor. Podemos ignorar, exagerar o mantener real el atenuamiento por distancia mediante la variación del factor. Este factor puede variar de DS3D_MINROLLOFFFACTOR (0) a DS3D_MAXROLLOFFFACTOR (10.0). EL valor DS3D_MINROLLOFFFACTOR significa que no se aplicará ninguna atenuación por distancia en el sonido, y cualquier otro valor indica un múltiplo del efecto en el mundo real.



En los juegos tridimensionales la importancia del sonido posicional es maxima.



Jedi Knight es un buen ejemplo de videojuego con sonido posicional.

un vector de posición (3.5, 2.0, -3.0) significa que el objeto está 3.0 metros a la derecha, 2.0 metros por encima y 3.0 metros por detrás del origen. Si el factor de distancia es cambiado a 2.0, este mismo vector significa que el objeto está a 7 metros a la derecha, 4 metros por encima y 6 metros por detrás del origen.

ESTABLECIENDO EL FACTOR DOPPLER

Para establecer o consultar el valor del factor *Doppler* usaremos los métodos **IDirectSound3DListener::SetDopplerFactor** e **IDirectSound3DListener::GetDopplerFactor**. El efecto *Doppler* aplicado a los sonidos respecto a su velocidad puede ser ignorado, atenuado o exagerado mediante el correcto establecimiento del factor *Doppler*. Asimismo, también puede ser establecido para imitar al efecto en el mundo real, mediante su valor por defecto.

Hay un método que nos permite cambiar o cunsultar todos sus parámetros de una sola llamada

El factor *Doppler* puede variar entre los valores *DS3D_MINDOPPLERFACTOR* y *DS3D_MAXDOPPLERFACTOR*, actualmente definidos como 0.0 y 10.0 respectivamente. Un valor de 0 significa que no se aplicará el efecto *Doppler* a ningún sonido, y cualquier otro valor representa un múltiplo del valor del efecto en el mundo real. Esto implica que un valor de 1 en el factor (o *DS3D_DEFAULTDO PPLERFACTOR*) significa que el efecto *Doppler* calculado será idéntico que en el mundo real, y un valor de 2.0 significa que el efecto se duplica.

ESTABLECIENDO LA POSICIÓN Y VELOCIDAD DEL OYENTE

Podemos gestionar la posición espacial del oyente mediante los métodos **IDirectSound3DListener::SetPosition** e **IDirectSound3DListener::GetPosition**. Para establecer o consultar el valor de velocidad usado para realizar los cálculos del efecto *Doppler* usaremos los métodos **IDirectSound3DListener::SetVelocity** e **IDirectSound3DListener::GetVelocity**.

ESTABLECIENDO LA ORIENTACIÓN DEL OYENTE

Para cambiar la orientación del oyente, definida por sus vectores superior y frontal, podemos invocar el método **IDirectSound3DListener::SetOrientation**, y para consultarla llamaremos al método **IDirectSound3DListener::GetOrientation**. El vector superior por defecto es (0, 1.0, 0) y el vector frontal por defecto es (0, 0, 1.0).

GESTIÓN SIMULTÁNEA DE TODOS LOS PARÁMETROS

Las interfaces **IDirectSound3DListener** e **IDirectSound3DBuffer** poseen cada una un método muy útil, pues nos permite cambiar o consultar todos los parámetros que tenemos de una sola llamada. Estos métodos son los siguientes: **IDirectSound3DListener::GetAllParameters** e **IDirectSound3DListener::SetAllParameters** para la interfaz **IDirectSound3DListener**, y **IDirectSound3DBuffer::GetAllParameters** e **IDirectSound3DBuffer::SetAllParameters** para la interfaz **IDirectSound3DBuffer**.

Autor: Carlos Gzlez. Morcillo

Curso de animación [IV]

Para poder practicar realizando los ejemplos del curso de animación, os explicamos las funciones básicas de Autodesk Animator Pro (quizás el programa de animación 2D más utilizado en el mundo de los videojuegos) que, pese a su "edad", se sigue manteniendo en las primeras posiciones como herramienta preferida de animadores al estilo tradicional.

En esta cuarta entrega de nuestro curso, nos introduciremos levemente en los intrincados mundos de Autodesk Animator Pro. Las posibilidades del programa son muchas, y sólo nos centraremos en las que sean de interés para desarrolladores de videojuegos. De esta forma, nos "saltaremos" algunas opciones como las del menú *optics* (efectos especiales de animator para mover un cel o flic por la pantalla) y daremos más importancia a algunas otras (como la manipulación de la paleta gráfica y los efectos de retoque).

Hemos recibido algunas cartas y mails de los lectores preguntando: "¿Y los que no tengan Animator?". Hay muchos programas de animación 2D en el mercado y todos se basan más o menos en los mismos principios. Aunque tengan otro interfaz de usuario y algunas opciones más, lo explicado en este artículo será extrapolable a cualquier programa destinado a la animación tradicional.

COMENZAMOS...

Al arrancar el programa, nos encontramos con 3 zonas diferenciadas en la pantalla; una barra de menús superior, el área de trabajo (donde dibujaremos a los personajes) y la zona de paneles (que está inicialmente en la zona inferior de la pantalla pero que podremos desplazar libremente).

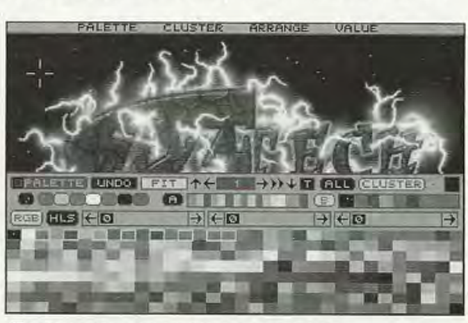
El *panel principal*: en él podemos distinguir varias zonas; a la izquierda, la zona de las herramientas (*drawing tools*). En el centro del panel principal está la mini paleta con los colores que estamos utilizando actualmente, el control de *frames* (para avanzar, retroceder, visualizar la animación completa...) y el degradado que hemos seleccionado en la paleta general (que usaremos con los modos *H Grad, L Grad, R Grad* y *V Grad*). A la derecha, encontramos los botones de modos o efectos (opaco, transparente, difuminados...).





Pantalla principal de AA Pro.

El *control de frames*: desde aquí podremos insertar, borrar y desplazarnos por los *frames* (cuadros individuales que componen nuestra animación). El número que sale dentro del recuadro central indica el número del *frame* en el que nos encontramos. Si pinchamos sobre la flecha arriba o abajo iremos al primer *frame* de la animación o al último respectivamente. Si pinchamos en la flecha hacia atrás o hacia delante, retrocederemos o avanzaremos un *frame* respectivamente. La doble flecha apuntando hacia la derecha sirve para ver la animación completa. Pinchando con el botón derecho sobre esta parte del panel principal, accederemos a un segundo menú; desde este segundo menú podremos insertar o borrar uno o más frames. Si pinchamos sobre una de esos dos botones (*INSERT/DELETE*) con el botón derecho tendremos la posibilidad de insertar o



Trabajando con la paleta de colores.



Partes del panel principal de AA Pro.

borrar un conjunto de *frames*. Si pinchamos con el izquierdo sólo realizaremos la operación sobre un *frame*. En la parte inferior de este menú tenemos una barra de desplazamiento, *PLAY SPEED*. Aquí indicaremos la velocidad con que se pasará de un *frame* a otro (cuanto mayor sea el valor, más lento será el movimiento). Por defecto toma un valor de 5, que corresponde a 15 frames por segundo.

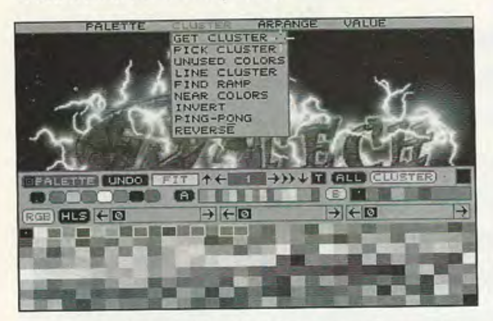
La *paleta*: es uno de los puntos fuertes de este programa, por lo que se ha ganado desde siempre un lugar privilegiado entre los grupos desarrolladores de videojuegos. Autodesk Animator Pro nos permite un control total sobre la paleta de colores de nuestra animación, por lo que podremos definir una única paleta de varios *frames* que usan paletas distintas.

Para editar un color de la paleta, bastará con que pinchemos sobre él y modifiquemos sus valores en la escala RGB (Rojo, Verde y Azul), con las barras de desplazamiento (que toman valores de 0 a 63). De esta forma, podremos definir los colores individualmente.

Una opción muy útil en este menú es la generación de rampas de color. Una selección de un conjunto de colores es un *cluster*. Con *Cluster/Get Cluster* podremos seleccionar un conjunto de colores, y con la opción *Value/Ramp*, indicándole un color de origen y uno de destino (del *cluster* activo) podremos crear rampas de color personalizadas.

Si tenemos un conjunto de *frames* en nuestra animación y cada uno tiene una paleta distinta (pero utilizan prácticamente los mismos colores), podremos recurrir a la opción

Palette/One Palette , que reducirá todas las paletas a una única para el conjunto de *frames*. Esto es muy útil en el método que vamos a seguir para los ejemplos de este curso; ya que importamos los *frames* de Adobe Photoshop y cada *frame* tiene una paleta.

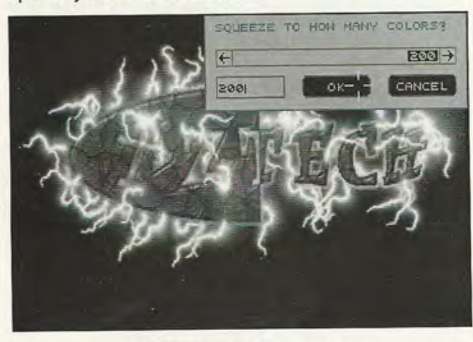


La opción *Palette/Files/Load* nos permite cargar una paleta, a la cual se adaptará el *frame* actual. Utilizaremos esta opción para que todas las animaciones de nuestro juego utilicen la misma paleta (si el juego va a ser en 256 colores, como los elaborados en DIV), repitiendo el proceso con cada *frame* de nuestras animaciones.



HEMOS SELECCIONADO UN CLUSTER.

Con *Value/Squeeze* podremos reducir la paleta de un *frame* al número de colores que queramos. Por ejemplo, si queremos reducir el número de colores de un *frame* a 200, tendremos que seleccionar como *cluster* toda la paleta (*start* en 0 y *end* en 255), y luego ir a *Value/Squeeze* e indicar que queremos reducir a 200. Realmente, si salvamos la imagen con una paleta que ha sufrido "Squeeze" se salvará con 256, pero sólo se usarán el número de colores que hayamos indicado.



REDUCIENDO LOS COLORES DE LA IMAGEN.

Los pasos que seguiremos en la creación de una paleta para nuestro juego serán:

- Editaremos los colores que necesitemos (en grupos de 16 en 16, por ejemplo) y a partir de ellos las "rampas de color".
- Salvaremos la paleta obtenida para aplicarla a todos los gráficos del juego.
- Al cargar gráficos de otros programas, adaptaremos sus paletas a la que tenemos salvada.

DISTINTAS HERRAMIENTAS DE AUTODESK ANIMATOR PRO

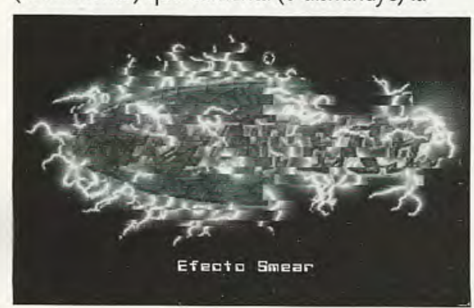
HERRAMIENTA	ACCION QUE REALIZA
Box	Dibuja formas rectangulares. Se definirá el rectángulo con 2 pulsaciones de ratón correspondientes a 2 esquinas opuestas. Admite modificaciones de tipo FILLED y 2COLOR
Circle	Dibuja formas circulares. En la primera pulsación de ratón se fijará el centro del círculo. Admite modificadores FILLED y 2COLOR
Draw	Dibujo a mano alzada. Para dibujar tenemos que dejar pulsado el botón izquierdo. Admite diferentes grosores.
Driz	Dibujo a mano alzada respecto de la velocidad a la que se mueva el ratón. Cuando la velocidad a la que movemos el ratón es más alta, el grosor de la línea disminuye. Cuando movemos el ratón lentamente, el grosor será el seleccionado.
Edge	Dibuja una línea de un pixel de grosor sobre la imagen actual. Seleccionaremos el color de la línea de contorno (en el cuadrado de color seleccionado del panel principal) y el color de fondo (en el cuadrado de color de fondo del panel principal) que será ignorado por Edge.
Fill	Cambia el color del pixel seleccionado y el de los adyacentes que tengan el mismo color. Su efecto será el de rellenar un área con un mismo color.
Fillto	Rellena áreas limitadas de pantalla con un borde de otro color.
Gel	Dibuja líneas con el borde difuminado (transparente, con degradación de color). El efecto conseguido dependerá del modo de dibujo que escojamos.
Line	Dibuja líneas rectas. Con dos pulsaciones de ratón definiremos una línea.
Move	Mueve la parte del dibujo seleccionada en un área rectangular. Con dos pulsaciones de ratón definiremos el área. Pulsando dentro del área podremos desplazar el dibujo seleccionado.
Oval	Dibuja formas ovaladas. En la primera pulsación se definirá el eje menor y en la siguiente el eje mayor. Admite modificadores FILLED y 2COLOR
Petal	Simula flores con pétalos simétricos respecto del centro. En la primera pulsación definimos el centro y en la segunda el radio. Se pueden definir el número de pétalos (2 a 32) y admite modificadores FILLED y 2COLOR
Poly	Dibuja formas poligonales. Admite modificadores FILLED y 2COLOR.
Rpoly	Dibuja polígonos regulares. En la primera pulsación se fijará el centro y en la segunda el radio. Admite modificadores FILLED y 2COLOR. El número de lados irá de 2 a 32.
Sep	Reemplaza un color por otro seleccionado desde la pantalla o la paleta.
Shape	Crea una figura cerrada dibujada a mano alzada. Al soltar el botón, la figura se cerrará automáticamente. Admite modificadores FILLED y 2COLOR
Spiral	Dibuja una espiral. En la primera pulsación definiremos el centro, en la segunda el radio, y girando con el ratón alrededor del centro iremos añadiéndole giros a la espiral. Para terminar, pulsaremos de nuevo el botón izquierdo del ratón.
Spline	Se utiliza para dibujar líneas curvas. Se pueden hacer cerradas o abiertas, admiten modificadores de tensión entre puntos, continuidad y curvatura.
Spray	Simula la utilización de un spray. Admite modificadores de velocidad de salida del aire y anchura de los puntos.
Star	Dibuja estrellas. Con la primera pulsación se fijará el centro y con la segunda el radio. Admite modificadores FILLED y 2COLOR. El número de puntas irá de 3 a 32.
Streak	Dependiendo de la velocidad con que se mueva el ratón dibujará líneas completas (si se mueve lentamente) o puntos separados.
Text	Escribe texto con el color seleccionado y la fuente elegida en FONT.



La mini paleta del panel principal contiene los colores más utilizados. Para cambiar estos colores, bastará con que pulsemos con el botón derecho sobre uno de ellos y acto seguido sobre un punto de la imagen o de la paleta general. Para seleccionar un color, bastará con pinchar con el botón izquierdo sobre él.



Las *herramientas*: seleccionando una u otra tendremos acceso a dibujar líneas, cajas, rellenar superficies... (ver tablas).
Podremos elegir de una extensa lista de distintas herramientas pulsando con el botón derecho encima de la zona de herramientas del panel principal o bien accediendo a *ani/tools*. Las herramientas que se utilizan para construir figuras rellenas (cajas, círculos...) suelen tener 2 opciones, *FILLED* y *2COLOR*. Con la primera se consigue que la figura se rellene del color activo, y con *2COLOR* la figura quedará hueca (sólo el borde saldrá del color seleccionado). Ambas opciones pueden seleccionarse a la vez.
Los *modos*: al igual que con las herramientas, también podremos seleccionar en una larga lista de modos de dibujo, si pinchamos con el botón derecho dentro del área asignada a los modos (*inks*) en el panel principal o bien en *ani/ink*. Los efectos básicamente admiten 2 opciones, *INK STRENGTH* (intensidad del efecto) y *DITHER* (movimiento) que aumenta (o disminuye) la





intensidad de un efecto y añade un efecto de imagen desenfocada respectivamente. Para hacernos una idea de cómo trabajan los distintos modos, una buena práctica sería ir probándolos uno a uno con la herramienta *box* (opción *filled* seleccionada) sobre una imagen que hayamos creado.
Los *Cel*: Un *cel* en animator es una parte de una imagen (o una imagen entera). Son de gran utilidad para desplazar trozos de una pantalla (personajes, *tiles*...), copiarlos y pegarlos en otra posición...
Dentro del menú *cel* encontramos varias opciones:

- *Clip*: selecciona como *cel* actual la parte del dibujo que no es el color definido como transparente.
- *Get*: podremos coger la parte del dibujo que queramos mediante un recuadro.
- *Move*: desplaza el *cel* por la pantalla sin pegarlo.
- *Paste*: "pega" *cel* en la posición de pantalla elegida.
- *Stretch*: podemos estrechar o ensanchar el *cel* actual.
- *Turn*: rota el *cel* un número de grados definido por el usuario.

Menú *Extra*: una opción muy interesante de este menú que nos permitirá automatizar tareas es la de definición de macros. Para tareas repetitivas que realicemos exactamente igual en varios *frames*, podemos definir macros que las hagan por nosotros. Antes de realizar la tarea a repetir, iremos a *Extra/Record/Start Record*. Hecho esto, realizaremos una repetición de la tarea (como pintar dos pixeles en blanco en el centro de la pantalla), avanzaremos un *frame* (¡importante!, de otra forma nos quedaríamos siempre en el *frame* actual) y pincharemos en *Extra/Record/End*



EL MENU CEL.

Record. Podremos repetir esta tarea de forma automática cuando queramos si vamos a *Extra/Record/Repeat* Macro (para repetir en varios *frames*) o en *Extra/Record/Use Macro* (para utilizar la macro en un único *frame*).





ALGUNOS ATAJOS DE TECLADO

Tecla	Función
@	Acceso rápido al panel de la paleta.
Z	Zoom
X	Borra la imagen actual
TAB	Selecciona el *cel* (recordad ajustar el color considerado como transparente)
M	Mueve el *cel* seleccionado (sin pegarlo)
'	Pega el *cel* seleccionado (pulsando después una flecha)
Cursor Izquierdo	Retrocedemos un *frame* en nuestra animación
Cursor Derecho	Avanzamos un *frame* en nuestra animación
Cursor Abajo	Visualiza la animación
Cursor Arriba	Nos situamos en el primer *frame* de la animación
B	Cambia el grosor del pincel

OBSERVACIONES

La intención de este artículo era introducir brevemente al lector en el manejo del Autodesk Animator, sin profundizar en él. Muchísimas cosas se han quedado en el tintero, pero creemos que no es tan importante explicar un programa a fondo nuestro curso de animación (ya que puede ser que utilicéis otros), como aprender los conceptos de animación. Este artículo ha sido escrito por petición de varios



lectores de Game Developer; ya que en los propósitos iniciales del curso no entraba explicar la utilización del programa. Practicando un poco con el programa, podéis averiguar sin problemas el resto de opciones no comentadas aquí. De cualquier forma, el «buzón» de esta sección (cgonmor@jet.es) queda abierto a todas y cada una de vuestras preguntas , sugerencias, y observaciones. Si no sugerís lo contrario, ya no se explicará nada más sobre el uso de programas (o más bien cosas puntuales). En el resto de entregas del curso nos centraremos exclusivamente en técnicas y teoría de animación (lo realmente importante). Un saludo, hasta el mes que viene.

MODOS DE DIBUJO (O EFECTOS)

MODO	EFECTO QUE CONSIGUE
Add	Añade el número de registro del color actual al número del registro del color de pantalla, aplicando a éste el color resultante de sumar ambos. Por ejemplo, si el color seleccionado es el 20 y el de pantalla es el 13, el nuevo color de pantalla sería el 33.
And	Realiza una función AND lógica con los colores (con el número del registro de la paleta del color seleccionado).
Bright	Incrementa la luminosidad del color. Admite modificadores INK STRENGTH y DITHER.
Close	Cierra huecos que queden en líneas definidas con el color actual.
Dark	Efecto "complementario" a *Bright*. Oscurece el color. Admite modificadores INK STRENGTH y DITHER.
Emboss	Simula un efecto de relieve en el dibujo; es decir, oscurece un borde del mismo y da luminosidad en otro. Admite modificadores INK STRENGTH y DITHER.
Glass	El color aplicado se vuelve transparente. Admite modificadores INK STRENGTH y DITHER.
Glaze	Idéntico al efecto GIASS, pero únicamente está activo cuando el botón del ratón está pulsado. También admite modificadores INK STRENGTH y DITHER.
Glow	Realiza un ciclo de colores, cambiando los de la pantalla de acuerdo con los seleccionados en el degradado del panel principal.
Gray	Reduce la saturación de los colores; los colores van pasando progresivamente a tonos grises. Admite modificadores INK STRENGTH y DITHER.
H Grad	Degradación Horizontal del degradado seleccionado en el panel principal. Admite modificador DITHER (con él seleccionado, el degradado no será lineal, consiguiendo un efecto más suave).
Hollow	Produce un contorno sólido al dibujo, reemplazando el color de fondo por el color seleccionado actualmente.
Jumble	Mezcla los pixeles del dibujo aleatoriamente. Admite modificador INK STRENGTH, que en este efecto denota el grado con que se mezclarán los mismos. Con un valor bajo de este parámetro se observará mejor el efecto.
L Grad	Degradación Lineal. Dibuja línea de contorno para algunas herramientas como *Circle* o *Gel*. Por lo demás se comporta igual que H Grad. Admite modificador DITHER.
Minus	Resta al color de la imagen el color seleccionado actualmente.
Opaque	El color aplicado será opaco.
Or	Realiza una función OR lógica con los colores (con el número del registro de la paleta del color seleccionado).
Pull	Mueve los pixeles de la imagen en la dirección que estamos pintando con el ratón. Es similar a la opción de "pintar con el dedo" de Photoshop. Para ver bien el efecto, utilizar la herramienta *Gel*.
R Grad	Degradación Radial. Permite situar el centro de la degradación y admite modificador DITHER.
Scrape	Simula una raspadura sobre el dibujo.
Slice	Mueve líneas verticales de pixeles en direcciones opuestas. Es similar al efecto *Split*, pero en vertical. También admite modificador INK STRENGTH.
Smear	Desplaza los pixeles hacia la dirección en que movamos el ratón. Es parecido a *Pull*, con la diferencia que *Smear* no "ensucia" el dibujo.
Smooth	Efecto parecido a *Soften*. Realiza el desenfoque de la imagen desplazando ligeramente los pixeles. Admite modificador DITHER.
Soften	Crea un efecto de desenfoque en la imagen. Es similar a la opción *Blur* de Photoshop. Admite modificador DITHER.
Spark	El píxel a colorear tomará el color resultante de sumar sus 4 pixeles adyacentes y dividir su color por 256. Si la paleta está "ordenada", es decir, los colores van por degradados ascendentes, dará el efecto de brillo sobre la imagen.
Split	Mueve líneas horizontales de pixeles en direcciones opuestas. Admite modificador INK STRENGTH que indicará la distancia que se separarán las líneas.
Sweep	Borra pixeles que estén aislados.
Tile	Repite el *cel* activo en la zona que se aplique el efecto. Admite modificador DITHER.
Unbuzz	Mezcla los pixeles, produciendo a la vez un cierto desenfoque. Reduce el parpadeo de las imágenes entrelazadas. Admite modificadores DITHER e INK STRENGTH.
Unzag	Desenfoque en modo de zigzag. Es especialmente útil para reducir el dentado de imágenes ya que, en rectas verticales u horizontales no realiza ningún efecto. Admite modificador DITHER.
V Grad	Degradación Vertical. Admite modificador DITHER.
XOR	Realiza una función XOR lógica con los colores (con el número del registro de la paleta del color seleccionado). Es decir, pintará con el color 0 cuando pintemos sobre una zona que tenga el color seleccionado y viceversa.